



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/739,618	12/18/2000	John H. Howard	5181-59100	3682

7590 12/10/2004

LAWERENCE J. MERKEL  
CONLEY, ROSE, & TAYON, P.C.  
P.O. BOX 398  
AUSTIN,, TX 78767-0398

EXAMINER

KIANERSI, MITRA

ART UNIT	PAPER NUMBER
----------	--------------

2145

DATE MAILED: 12/10/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	<b>Application No.</b> 09/739,618	<b>Applicant(s)</b> HOWARD, JOHN H.	
	<b>Examiner</b> mitra kianersi	<b>Art Unit</b> 2143	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

#### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 03 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) ☒ Responsive to communication(s) filed on 18 December 2000.
- 2a) ☒ This action is **FINAL**.                      2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) ☒ Claim(s) 1-43 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-43 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 18 December 2000 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All    b) ☐ Some \*    c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

- |                                                                                                                                   |                                                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)                                                       | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)                                              | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)             |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)<br>Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____                                                |

## Response to Arguments

Applicant's argument filed on June 21/2004 has been fully considered, but they are not persuasive.

Applicant on page 12, line 27 and page 13, lines 1-6 argue that the teachings include no discussion of inodes, nor of performing updates atomically. Moor in col 1, lines 36-41 disclose that users update the database data sets in the database, the database management system records the updates into a log data set. The log data set is an amount of data, such as a file, which reflects a series of updates to the database. Log data sets are recorded in sequential records, which have defined open and close points. Moor also in col 2, lines 10-27 discloses that in order to create the CADS, the change accumulation utility reads log data sets sequentially, that is, one after another. Typically, users organize their multiple databases into change accumulation groups so that the change accumulation utility operates as efficiently as possible. A user can run the change accumulation process against one change accumulation group and use an optional secondary output--the set of log records that were not written to the change accumulation data set--as input to the change accumulation utility for the next change accumulation group to be processed. This can be done for each change accumulation group in which the current change accumulation run uses the secondary output of the previous change accumulation run. This serial process is managed directly by the user. Users usually run change accumulation periodically so that when a database data set in a change accumulation group requires recovery, the time required to run a final change accumulation job and subsequent database recovery job is minimized. Moor in col 2, lines 30-37 discloses that the recovery utility reads the entire CADS into memory and applies that portion of the CADS that is relevant to the database being restored. Each record has an identification that's sequential and the database data sets are restored in a sequential order. The recovery utility addresses each record in the CADS to see if there is a change in data for that record. If so, the CADS is accessed and the relevant record merged into the new database.

Art Unit: 2143

Moor in col 10, lines 60-67 and col 11, lines 1-3 discloses FIG. 5 a sequence of method in step 500. Prior to initiation of this method one or more databases 206 have failed. The recovery method initiates in step 502. Initiation may include preparing the database recovery utility for operation, for example, by creating a separate address space to manage backup data sets, CADSSs, and log data sets, performing internal system checks, initializing memory and devices of required addresses, etc. Commands for implementing recovery may be executed by the database recovery utility 300 shown in FIG. 3.

Moor in col 7, lines 47-50 discloses the database system 200 further includes one or more databases 206 having one or more database data sets. The databases 206 are designated as DB1 to DBn to illustrate a variance in the number of databases 206 in a system 200.

Applicant on page 14, line 2, argue that there is no discussion of inodes, atomic updates, inode files, etc. storage and on specification explains that the storage may employ a transactional interface in which the various updates to a file performed between the opening of the file and the closing of the file are either committed as a whole to non-volatile storage or abandoned as a group. The storage may employ a copy on write protocol in which a block of a file to be updated is copied to a newly allocated block, and the newly allocated block is updated. The inode for the file (which is identified by the 1-number and indicates one or more blocks storing the file data) may be copied to a working copy of the inode, and the working copy may be updated to indicate the newly allocated blocks. In response to a file commit (e.g. a file close or a file synchronization command), the working copy of the inode may be atomically written to a non-volatile storage. In this manner, the updates performed during the transaction may be atomically committed to the file. Since the updates are atomically committed, the likelihood of file corruption in the case of a system failure may be reduced. Thus, reliability of the storage may be increased. Moor also in col 1, lines 36-41 disclose that users update the database data sets in the database, the database management system records the updates into a log data set. The log data set is an amount of data,

such as a file, which reflects a series of updates to the database. Log data sets are recorded in sequential records, which have defined open and close points.

Applicant on page 14, lines 1-2 and lines 24-25 argues that there is no discussion of inodes, atomic updates, inode files, etc. Moor in col 7, lines 56-65 discloses that each database management system 202 may include a log 204 having log records to track updates to data kept in memory 18 or in a database 206. The log 204 is used for reference to track data changes and other events performed by the corresponding database management system 202. Changes and other events are stored on the log 204 as log records. The log 204 may be stored on one or more memory devices 18 of the station 12.

Moor in col 11, lines 8-12 discloses that in step 504, the recovery utility 300 builds a recovery list, which is a collection of databases 206 to be recovered. In one embodiment, when a recovery list is built in step 504, it is associated with a logical terminal that issued the recovery command. Moor in col 11, lines 13-23 discloses that recovery continues in step 506 when the recovery utility 102 receives a command to start the recovery. The recovery utility 300 performs a check to determine if recovery is currently in process or if a desired recovery list cannot be found. If so, an error message issues and recovery is aborted. Otherwise, recovery continues. The recovery utility 300 validates the recovery list by ensuring that each database 206 is in a state that allows it to be recovered, and also determines the resources needed for recovery of these validated entries.

Regarding claims 4, 9, 19, and 24 and claims 5, 10, 20, and 25 and their inherency as a file close and as fsync command, Moor in col 11, lines 1-3, discloses that Commands for implementing recovery may be executed by the database recovery utility 300 shown in FIG. 3. Once the initiation step 502 commences, the remaining steps of the method 500 are performed automatically without user intervention. Also, in col 10, lines 12-26, discloses that the database recovery utility 300 further comprises an image copy and restore utility 316 which serves to create the restored databases 318. The image copy and restore utility 316 receives the backup copy 210 from the backup copy restore utility 302 and uses the backup copy 210 as a basis for creating one or more restored

databases 318. The image copy and restore utility 316 further receives data sets from the CADS manager 304. The image copy and restore utility 316 coordinates (corresponds to synchronizing) application of the data sets from the CADS 214 in an appropriate sequential order to create a restored database 318. After the image copy and restore utility has created and written to the restored databases 318, the database update manager 314 merges the log data sets into the restored databases 318 in the appropriate location. Because the arguments with respect to the allowableness of independent claims were found unpersuasive, these same arguments are not persuasive with respect to the other dependent claims.

Claims 1-43 have been examined.

### ***Claim Rejections - 35 USC § 102***

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language:

Claims 1-35 are rejected under 35 U.S.C. 102(e) as being anticipated by Moore (US Patent No. 6,594,676).

1. As per claim 1, a storage comprising: a non-volatile memory storing a first inode corresponding to a first file; (the memory devices 18 are depicted as including a non-volatile storage device 20 such as a hard disk drive, CD ROM drive, tape drive, or any other suitable storage device, col 6, lines 39-41) and a block manager configured to copy said first inode to a second inode, wherein said block manager is

configured to change said second inode in response to updates to said first file, and wherein said block manager is configured to atomically update said first file in response to a commit of said first file by writing said second inode to said non-volatile memory. (One method for implementing database updates and commit point processing is for the database manager to maintain the database changes in storage and not apply the changes to the databases until the commit point is reached. A copy of the database data that is changed is written to the log as the update is created. When the commit point is reached, and everything went as expected, the updates are written to the databases, col 2, lines 52-60).

2. As per claims 2 and 22, the storage wherein said non-volatile memory stores a journal comprising a list of committed inodes, and wherein said block manager is configured to record said second inode in said journal. (a copy of the database data that is changed is written to the log as the update is created. When the commit point is reached, and everything went as expected, the updates are written to the databases, col 2, lines 55-57).

3. As per claims 3 and 23, the storage wherein said commit of said first file comprises a commit commands received from an external source which updates said first file. (When the commit point is reached, and everything went as expected, the updates are written to the databases, col 2, lines 57-59).

4. As per claim 4, 9,19 and 24, the storage wherein said commit command comprises a file close command. (The step is inherent, because among the library routines may be routines to open a file, read a file, write a file and close a file).

5. As per claim 5, 10, 20 and 25, the storage wherein said commit command comprises an fsync command. (The step is inherent, because a synchronization command like Unix fsync command may be supported which may commit all prior changes without closing the file).

6. As per claims 6 and 26, the storage wherein said journal further includes a checkpoint record including a description of an inode file, a block allocation bitmap, and an inode allocation bitmap. (each database management system 202 may include a log 204 having log records to track updates to data kept in memory 18 or in a database 206. The log 204 is used for reference to track data changes and other events performed by the corresponding database management system 202, col 7, lines 39-43).

7. As per claims 7 and 27, the storage wherein the description comprises inodes for each of said inode file, said block allocation bitmap, and said inode allocation bitmap. (the database system 200 further includes one or more databases 206 having one or more database data sets. The databases 206 are designated as DB1 to DBn to illustrate a variance in the number of databases 206 in a system 200, col 7, lines 47-50).

8. As per claim 8, an apparatus comprising: a computing node configured to perform one or more write commands to a file and a commit command committing the one or more write commands to said file; and a storage coupled to receive said one or more write commands and said commit command, wherein said storage is configured to copy one or more blocks of said file to a copied one or more blocks, said one or more blocks updated by said one or more write commands, and wherein said storage is configured to update said copied one or more blocks with write data corresponding to said one or more write commands, and wherein said storage is configured to copy a first inode corresponding to said file to a second inode and to update pointers within said second inode corresponding to said one or more blocks to point to said copied one or more blocks, and wherein said storage is configured to atomically update said file by writing said second inode responsive to said commit command, and wherein said first inode is stored in an inode file, and wherein said inode file is identified by a master inode, and wherein said inode file is atomically updated with said second inode by writing said master inode subsequent to said commit command. (database management systems include a recovery facility to



respond to a database failure. Upon database failure, the recovery facility creates a new database and writes the backup copy to the new database. The recovery utility further applies all the updates to the database from when the backup copy was created. Information used to restore the new database from the last state of the backup copy may be taken from the log data sets and recovery control information, col 1, lines 57-65)

9. As per claims 11 and 28, a method comprising: copying a first inode corresponding to a first file to a second inode; modifying said second inode in response to one or more changes to said first file; and atomically updating said first file by establishing said second inode as the inode for said first file. In order to create the CADS, the change accumulation utility reads log data sets sequentially, that is, one after another. (typically, users organize their multiple databases into change accumulation groups so that the change accumulation utility operates as efficiently as possible. A user can run the change accumulation process against one change accumulation group and use an optional secondary output--the set of log records that were not written to the change accumulation data set--as input to the change accumulation utility for the next change accumulation group to be processed. This can be done for each change accumulation group in which the current change accumulation run uses the secondary output of the previous change accumulation run, col 2, lines 12-29).

10. As per claims 12 and 29, the method wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory. (a portion of the non-volatile storage 20 which is used to extend the RAM 24, col 6, lines 47-48).

11. As per claims 13 and 30, the method further comprising writing a master inode corresponding to an inode file including said second inode to a checkpoint record in said journal. the updates are not permanently stored in the database until the updates are physically written on the database. (In general, database activity is based on being able

Art Unit: 2143

to "commit" updates to a database. A commit point is a point in time where updates become permanent parts of the database, col 2, lines 43-46).

12. As per claims 14 and 31, the method wherein recovering from a system failure comprises: scanning said journal to locate a most recent checkpoint record and zero or more inodes subsequent to said most recent checkpoint record within said journal; copying said master inode from said most recent checkpoint record to a volatile memory; and updating an inode file corresponding to said master inode with said one or more inodes subsequent to said most recent checkpoint record. (In recovery, subsequent updates to the database are applied from records on the log data sets. Recovery further requires storage of attributes of the database and the backup. Database management systems often include a data set for control of recovery, which comprises several attributes of the database and the backup copy. Database management systems use some form of recovery control information recorded in this data set relating to the database and the backup copy to assist in recovery. Col 1, lines 43-56).

13. As per claims 15 and 32, the method wherein said updating said inode file comprises: copying one or more blocks of said inode file storing said one or more inodes to a copied one or more blocks; (If something goes wrong, such as a write error to the database, and the updates cannot be made, all the updates produced since the last commit point are "aborted." It is as if the updates never happened, col 2, lines 48-51) and updating said master inode in said volatile memory to point to said copied one or more blocks. for implementing database updates and commit point processing is for the database manager to maintain the database changes in storage and not apply the changes to the databases until the commit point is reached, col 2, lines 52-55).

Claims 16, 17, 33 and 34; recite the same limitations as claim 11. Therefore, it is analyzed and rejected by the same rationale.

14. As per claims 18 and 35, the method wherein said establishing said second inode is performed in response to a commit command. (a copy of the database data that is changed is written to the log as the update is created. When the commit point is reached, and everything went as expected, the updates are written to the databases, col 2, lines 55-57).

15. As per claim 21, a storage comprising: a non-volatile memory storing a first inode corresponding to a first version of a file; (the memory devices 18 are depicted as including a non-volatile storage device 20 such as a hard disk drive, CD ROM drive, tape drive, or any other suitable storage device, col 6, lines 39-41) and a block manager configured to copy said first inode to a second inode wherein said block manager is configured to change said second inode in response to updates to the file, and wherein said block manager is configured to atomically update the file, producing a second version of the file, in response to a commit of the file by writing said second inode to said nonvolatile memory. (One method for implementing database updates and commit point processing is for the database manager to maintain the database changes in storage and not apply the changes to the databases until the commit point is reached. A copy of the database data that is changed is written to the log as the update is created. When the commit point is reached, and everything went as expected, the updates are written to the databases. Col 2, lines 52-60).

### ***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 36-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over Moor (Us Patent. No. 6,594,676) and further in view of Baird et al. (Distributed information storage architecture).

16. As per claim 36, Moor discloses a storage comprising a non-volatile memory storing a first file. Moor fails to teach a block manager coupled to receive a plurality of write commands and a commit command from an interconnect to which the storage is coupled during use, wherein the plurality of write commands update the first file, and wherein the commit command is defined to commit the updates to the first file, wherein the block manager is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command. However, Baird et al. disclose a distributed information storage architecture having an information storage-management divides systems into the following major domains: information (intelligence applied to facts), data (raw un-interpreted facts, and storage (repository for data). For example applications use files; files use volumes and volumes use devices. In this example, an application (in the information domain) stores information in files. A file manager (in the data domain) stores files on volumes. A volume manager (in the storage domain keeps track of volumes and mounts volumes on devices (also in the storage domain) and a device driver schedules I/O requests. Therefore, it is obvious to one ordinary skill in the art to employ the distributed information storage architecture disclosed by Baird to the recovery utility apparatus disclosed by Moor because it would be an advancement to provide a simplified database recovery apparatus and systems that substantially reduces recovery time after database failure.

17. As per claim 37, the storage wherein the first file in the non-volatile memory is a first version of the first file, and wherein the block manager is configured to create a second version of the first file in response to a first write command of the plurality of write commands, and wherein the block manager is configured to atomically replace the first version with the second version in response to the commit command. (Fig.6 gives an example of a physical storage class hierarchy. Also, A storage policy manager regulates operations on storage objects based on policy options and rules. Policy managers can modify the behavior of storage objects by intercepting operations on storage objects and supplementing the objects' methods with methods of its own. For

example a space allocation manager might implement supplementary methods for the WRITE operations to a virtual volume. Page 152, [2], Baird et al.)

18. As per claim 38, the storage wherein the first version is associated with a first inode, and wherein the second version is created by copying the first inode to a second inode and modifying the second inode, and wherein the atomic update is performed by writing the second inode. (Data-management utilities (that move, copy, and otherwise manage data), encapsulation services (that hide internal encoding and location of variables), and conversion services (that transform streams of data from one format to another) are also part of this domain. Page 146, Baird et al. [9])

19. As per claim 39, the storage wherein the storage is an object-based storage and wherein the plurality of write commands and the commit command address a file object corresponding to the first file. (the storage domain supports the storing, retrieval and safekeeping of persistent data. This domain provides management and administration for logical storage object such as linear spaces. Virtual storage object such as virtual volumes, and physical storage object such as devices, page 146, Baird).

20. As per claim 40, an apparatus comprising:  
a computing node configured to generate a plurality of write commands to update a first file and a commit command defined to commit the updates to the first file;  
an interconnect to which the computing node is coupled, wherein the computing node is configured to transmit the plurality of write commands and the commit command on the interconnect; and a storage coupled to the interconnect and configured to store the first file, wherein the storage is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command. (An enterprise can configure several storage nodes (each with their own contingent of storage devices) into a single storage system. An administrator can add a storage node to a storage system or remove one without disrupting work on other storage nodes. The relationship between two storage nodes is illustrated in Figure 8. Baird et al.)

21. As per claim 41, the apparatus wherein the first file on the storage prior to the plurality of write commands is a first version of the first file, and wherein the storage is configured to create a second version of the first file in response to a first write command of the plurality of write commands, and wherein the storage is configured to atomically replace the first version with the second version in response to the commit command. (Data-management utilities (that move, copy, and otherwise manage data), encapsulation services (that hide internal encoding and location of variables), and conversion services (that transform streams of data from one format to another) are also part of this domain. Page 146, Baird et al. [9])

22. As per claim 42, the storage wherein the first version is associated with a first inode, and wherein the second version is created by copying the first inode to a second inode and modifying the second inode, and wherein the atomic update is performed by writing the second inode. (Data-management utilities (that move, copy, and otherwise manage data), encapsulation services (that hide internal encoding and location of variables), and conversion services (that transform streams of data from one format to another) are also part of this domain. Page 146, Baird et al. [9])

23. As per claim 43, the storage wherein the storage is an object-based storage and wherein the plurality of write commands and the commit command address a file object corresponding to the first file. (the storage domain supports the storing, retrieval and safekeeping of persistent data. This domain provides management and administration for logical storage object such as linear spaces. Virtual storage object such as virtual volumes, and physical storage object such as devices, page 146, Baird).

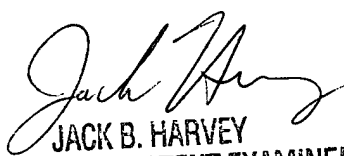
### ***Conclusion***

**THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mitra Kianersi whose telephone number is (571) 272-3915. The examiner can normally be reached on 7:00AM-4:00PM.

Mitra Kianersi  
Nov/22/2004

  
**JACK B. HARVEY**  
**SUPERVISORY PATENT EXAMINER**